`

Conditional Clustering Method on KNN for Big Data

A Thesis Presented in Partial Fulfillment of
Bachelor of Science
(Honors in Statistics)

Qi Chen

Supervisor: Prof. Mark Fredrickson

Final Version: May 23, 2022

Abstract

This thesis considers challenges faced by k-nearest neighbor (KNN) classifiers when handling big data, particularly concerning large storage requirements and extended training times. The proposed solution revolves around data filtering techniques. Drawing upon the fundamental principle of KNN classifiers, which assumes that similar data possess similar conditional distributions regarding the response variable, this study advocates for employing clustering methods to segment the training data and subsequently filter it by selecting the closest cluster as the training set. Stem from Bayes' rule and the Mixture distribution of data, the clustering refinement involves utilizing clustering techniques conditional on the class of responses. To execute this approach, the algorithm prefers a hierarchical clustering model, chosen for its stability and efficiency. To counterbalance the loss of information resulting from filtering the training set, the algorithm replaces the standard KNN classifier with a local KNN classifier. By locally adjusting the parameters of the KNN classifier, the model achieves a more favorable trade-off between bias and variance. The effectiveness of the proposed model is evaluated using three sets of real-world data: Fashion MNIST, Forest Cover Type Prediction, and Online Shoppers Intention from the UCI Machine Learning repository. The results of the tests demonstrate that the conditional clustering method significantly enhances runtime efficiency while employing the local KNN classifier improves model prediction ability. Notably, the number of clusters proves to be a critical factor influencing the model's accuracy. While increasing the number of clusters may reduce the filtered training dataset size, thus resulting in information loss, a higher number of clusters affords the local KNN classifier greater opportunities to strike a balance between variance and bias, consequently lowering model risk.

Contents

1 Introduction

The K-nearest neighbor (KNN) classifier stands out as a robust supervised machine learning algorithm suitable for classification tasks. As a non-parametric method, it does not rely on assumptions about the underlying data distribution but instead operates on an instance-based approach, in which it classifies new data points by comparing them with existing training data points (Hastie, Tibshirani, & Friedman, 2009, chapter 13).

The classification process in the KNN algorithm involves finding the K nearest neighbors for a given data point $x_t \in S^d$ in a metric space S^d, where k is an integer chosen by the researcher. Given a training set $X \in S^d$, to predict a class label or produce a regression estimate at x_t, k data points with the closest distance to the test point are found using a distance function. Let $S_x \in S^d$ be the k nearest neighbors of the test point:

$$S_x \subseteq X \mid \left|S_x\right| = k \wedge \forall x_1 \subseteq X, S_x \forall x_2 \subseteq S_x, dist(x_t, x_1) > dist(x_t, x_2)$$

These k-nearest points are then averaged to achieve a regression prediction $\hat{Y}$ as follows:

$$\hat{Y} = \frac{1}{k} \sum_{i \in S_x} Y_i .$$

Classification is usually done through a voting mechanism based on the class labels of the nearest neighbors. Since the voting mechanism is based on the belief that similar data has a similar distribution of the response, it is reasonable to allocate more weight to more similar data, which is similarity-weighted voting. Assume responses Y to be a categorical random variable with J levels, X is a collection of predictor values, and $\{(x_i, y_i)\}$ is a collection of training data, the simple voting and similarity-weighted voting for a test point y_j with predictor value x_j to be category j are given by (Cunningham & Delany, 2021):

Simple voting: $Vote(y_j = j) = \sum_{i=1}^{k} I(y_i = j)$

similarity weights voting: $Vote(y_j = j) = \sum_{i=1}^{k} \frac{1}{d(x_j, x_i)} I(y_j = j)$

While the KNN method stands out for its simplicity and intuitive nature, its reliance on calculating distances between the point x and all training samples makes it computationally expensive, especially as dataset sizes escalate. The inefficiency of the KNN method becomes more pronounced with big data due to several factors. One of the most important is that the computational complexity of the KNN method scales linearly with the size of the training dataset, resulting in longer processing times and increased memory requirements. The choice of distance function will also affect the calculation time, and many commonly used distance functions have computational complexity that scales much worse than linearly (Cunningham and Delany, 2021). For the KNN method to be a competitive technique in today's era of exponential data growth, additional modifications are necessary. To mitigate these challenges and improve the scalability of the k Nearest Neighbor method, I propose a novel method of integrating clustering as a pre-processing step. By partitioning the training data into distinct clusters based on similarity measures, clustering reduces the effective size of the dataset that the KNN method needs to operate on. The specific operation is only using training data within the closest cluster to make predictions instead of considering the entire dataset. While clustering has been used as a solution in previous adaptations of the KNN method, the approach of this paper forms clusters conditional on class labels. This approach improves both the statistical and computational performance of the KNN classifier.

1.1 Statistical Performance of the KNN Classifier

As the training sample size (n) increases, the KNN method can incorporate more data points for making predictions, so in principle, a larger value of k can be used. This typically leads to smoother decision boundaries and reduces the impact of noise in the data. But if k is allowed to grow too quickly relative to n, the variance of the estimates will not decrease. Stone has proved that when k and n approach infinity and k/n approaches 0, the KNN estimator is universally consistent, and the risk converges to the Bayes risk, the optimal prediction accuracy if the true conditional distributions were known (Stone, 1977). Since the proper value of k is rarely known in practice, selecting a value of k using the training data is of critical importance.

A frequently used method for selecting machine learning model parameters is k-fold cross-validation, where the parameter with the lowest cross-validation error is applied to the entire dataset for training. However, since the parameters of the KNN method are dependent on the training set size, leave-one-out cross-validation (LOOCV) is used to determine the optimal value of k. Azadkia (2020) has demonstrated that LOOCV identifies the optimal k quickly and efficiently through a non-asymptotic error bound.

The performance of the KNN classifier is heavily influenced by the data distribution. The KNN method operates on the belief that closely located data points exhibit similar conditional distributions of the response variable given predictors. Consequently, in regions where data is densely distributed, the model can access sufficiently similar data points, potentially leading to high accuracy. However, when test data is situated in sparse data distribution areas, the training set used for classification may lack similarity resulting in decreased prediction reliability.

Cannings et al. (2020) propose a method called local KNN which allows the choice of k for KNN classifiers to depend on the estimated density of x so that the estimation k value varies based on the estimation of the distribution of predictors. Cannings et al use a family of bandwidths {h(Xi) : i = 1,...,n} instead of h = h(x) as density estimation so that the resulting estimate is itself a density (Cannings et al., 2020).

Because the KNN classifier is an instance-based classifier, filtering data is an effective way to improve the training process while maintaining classification ability based on the training data.

One such method is the Condensed Nearest Neighbor (CNN) rule, proposed by P.E. Hart in 1968. This method preserves the essence of the nearest neighbor idea but uses the filtered consistent training subset. The CNN rule works by iteratively collecting data points that are misclassified by the model based on the existing collected data. By doing so, CNN filters a consistent subset by removing superfluous observations that will not affect the classification accuracy of the training set (Hart, 1968). These observations are often referred to as interior points since data points around the center of a cluster tend to have little effect on classification, while those around the boundary of a cluster are more likely to influence the classifier's decisions. Filtering a consistent training subset aims to select a subset of the training set that classifies the remaining data correctly through the nearest neighbor (NN) rule (Hart, 1968). However, the CNN method faces certain challenges: 1. Whether data is included in the training set depends on the order in which it is selected. 2. For highly overlapping data, CNN tends to select all data in the original training dataset.

Devi and Murty (2022) propose an MCNN rule (Modified CNN rule) dealing with CNN's order-dependent problem. The MCNN begins by defining a foundational set of prototypes, each representing a unique class. The training dataset is then classified using these initial prototypes. Following this, through the identification of misclassified samples, a representative prototype for each class is recognized and added to the original set. The expanded set of prototypes is subsequently utilized for another cycle of classification on the training data. This iterative process continues as representative prototypes are fine-tuned based on misclassifications, ultimately ensuring accurate classification of all patterns within the training set. By addressing order-dependent issues, the MCNN tends to establish a consistent training subset with less data than CNN, while still ensuring 100% classification accuracy on the training set.

As an instance-based classifier, the KNN classifier's parameter value is influenced by the number of observations. A larger training set provides more data similar to the test points, enabling the KNN classifier to better approximate Bayes' classifier and achieve higher accuracy. However, even without utilizing the entire training set, the clustering KNN methods can train a KNN classifier using a consistent training subset that achieves accuracy comparable to that of a KNN classifier trained on the full dataset if the clusters are chosen appropriately.

1.2 Computational Performance of KNN

The computational complexity of the search method in KNN increases with the size of the training dataset for each test sample, denoted as O(nd), where n is the size of the training dataset and d is the dimensionality (Deng et al., 2016). To illustrate this behavior, refer to Figure 1, which displays the relationship between the training time of KNN classifiers (with a k value of 10) and sample sizes. Similarly, Figure 2 shows the relationship between the training time of KNN classifiers (with a k value of 10) and the number of predictors. Observe that the training time is approximately linear in the size of the data set.

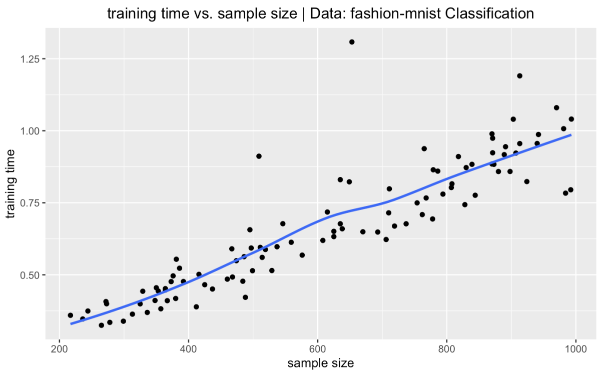


Figure 1

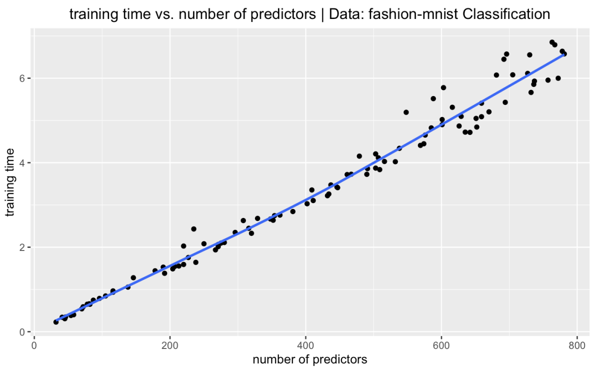


Figure 2

To improve the computational performance two main approaches are using better data structures to allow for faster distance computations and clustering to effectively reduce the dimensionality of the problem to a one-dimensional measure.

Filtering the training set is a practical approach to enhance the training efficiency of KNN. R-Tree, a specialized index designed for range queries, can quickly identify all entries either contained within or overlapping a specified query. Based on the problem of finding similar tumor shapes, Korn et al. proposed a faster Nearest Neighbor Search in 1996. The method mainly solves two problems: how to measure the distance between two shapes and how to do better than sequentially scanning the entire database given such a distance function. To solve the problems, Korn et al apply the F-index with n-d R-tree to the KNN method. Since not all data in the training set influences predictions and most data with low similarity to the test points are irrelevant to the prediction results, using n-d R-trees, which are tree data structures designed for spatial access methods, can significantly enhance the training efficiency of the KNN method. The specific operation is to submit the calculated size distribution of each image to a range or k-nearest neighbor to search in the R-tree. This method can complete the search faster without compromising correctness (Korn et al, 1996).

Clustering can also be a helpful technique to reduce the problem of sample size. Deng et al propose an LC-KNN (Landmark-based Spectral Clustering) method to improve the efficiency of the KNN method. Before doing nearest neighbor estimation, LC-KNN computes a Landmark-based Spectral Clustering which has two advantages: low complexity and scales linearly. The distinctive feature of Landmark-based Spectral Clustering lies in its representation of each sample through its projection onto a set of basis vectors. After clustering training data, LC-KNN uses training data in the closest cluster to make predictions. After clustering the training dataset, KNN can focus solely on the training data within the cluster nearest to the test point, which significantly reduces the computational load of the KNN method (Deng et al., 2016).

There are still some flaws in the design of LC-KNN. The shapes of the formed clusters are uncertain, which means the nearest cluster might not contain most of the training data that is similar to some test points. Saadatfar et al propose an improvement based on LC-KNN. In addition to considering the center of each cluster, their method also considers the shape and distribution under each cluster. They demonstrate that KNN is more efficient after considering those 2 parameters (Saadatfar, Khosravi, Hassannataj Joloudari, Mosavi, & Shamshirband, 2020).

2. Method

A common theme when considering both the statistical and computational performance of KNN is that clustering can be a useful method to help improve local estimation of optimal k and also reduce the search space when training and predicting using KNN. In this section, I will introduce the two steps of this method, namely creating conditional clusters and making predictions using the nearest neighbor method. The goal of the first part is to create clusters of similar data for each response variable class. The second part will use the nearest clusters as the training data set for prediction, which can reduce the usage of training data while maintaining the required data.

When employing the nearest neighbor method, we leverage a neighborhood distribution to approximate the distribution of the response variable at a specific point. However, this often leads to an excess of calculations, as numerous distances are computed solely to identify the nearest k neighbors.

Clustering, a common unsupervised learning technique, organizes data based on their similarities, making it a valuable tool in data preprocessing. Similar to the nearest neighbor method, clustering methods create groups of observations according to the data's similarity (Hastie et al., 2023). Consequently, clustering can effectively classify data in advance of applying KNN with any particular outcome Y. By properly creating clusters in the training set, when performing classification or regression, we can eliminate a large amount of data with low similarity to the predicted points, thereby reducing unnecessary calculations.

As previously discussed, preprocessing the data through clustering before applying nearest neighbor methods has been proposed, particularly as a method for local KNN techniques that select k = k(x), where the number of nearest neighbors is a function of the prediction point x. In lower-density areas of the predictors, clustering methods tend to create clusters including only a small number of training data. Cannings et al claimed that selecting k = k(x) allows for using fewer neighbors in low-density regions, leading to a more balanced trade-off between local bias and variance (Cannings et al., 2020). Consequently, clustering the data before using KNN classifiers intuitively reduces computational costs without significant information loss.

Clustering methods are reliant on the data distribution; hence, general clustering methods tend to form larger clusters centered around the densest regions of the sample space S^d. Figure 3 shows how the data can be divided into two clusters without knowing the classes of Y.

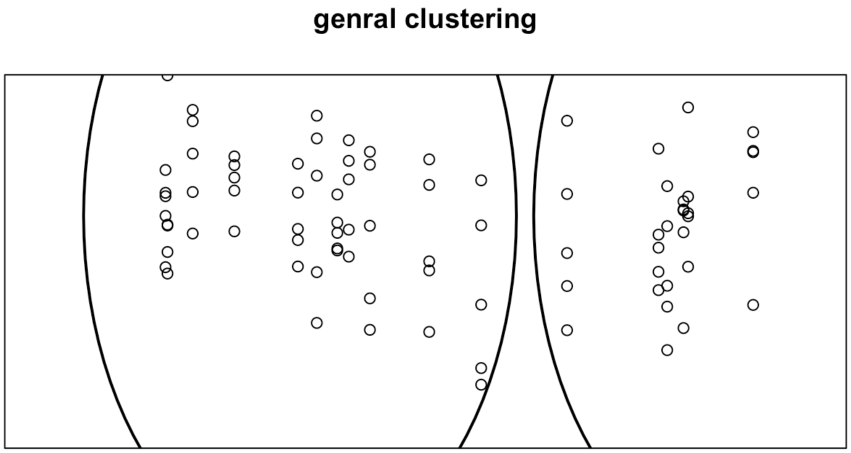


Figure 3

As an unsupervised technique, however, the clusters found when clustering the entire training set may not represent the clusters within a given response class. Therefore, performing clustering based on each class can help separate the mixed distribution of predictor values. Figure 4 shows how the data can be divided into two clusters for each class of Y.

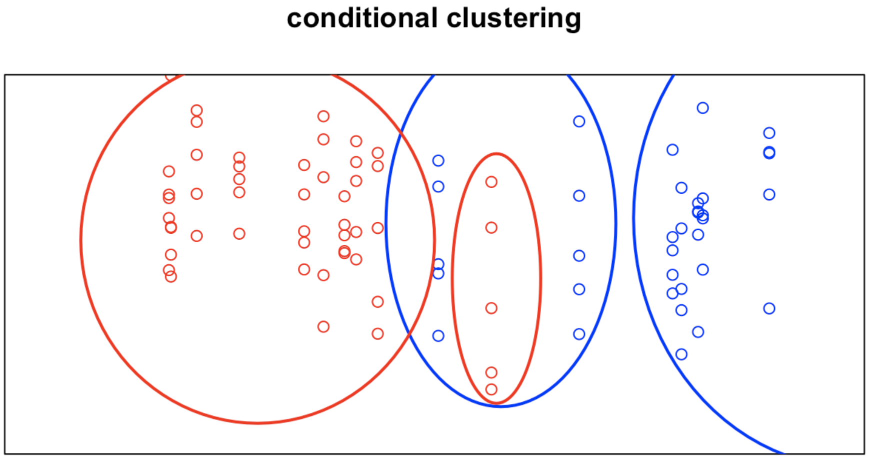


Figure 4

This thesis proposes an algorithm that creates clusters conditional on the response variable classes during the training phase. For prediction, it collects the closest cluster for each level of the response variable and performs KNN on the union of all clusters that contain the prediction point. Optionally, specific values of k can be learned for each set of overlapping clusters.

To further justify this approach, consider the application of Bayes' rule to predicting the probability of class label j.

$$P(Y = j|X = x) = \frac{f(x|Y=j)}{f(x)}$$

The marginal distributions of Y and X are easy to estimate, and unsupervised clustering can be used to approximate the marginal distribution of X, but it provides no information about f(x | Y = y), the conditional distribution of X for observations with class label j.

Therefore, the key point becomes estimating f(x|Y=y). Suppose this distribution can be written in the form of a mixture distribution composed of c clusters:

$$f(x \mid Y = j) = \sum_{i=1}^{c} \alpha_i f_i(x \mid Y = j)$$

where the α_i represent the marginal probability of falling into the conditional cluster i. If we further suppose that for any given value of x all but one $f_i(x \mid Y = j)$ approx 0, then estimating the marginal distribution of X given Y = j simplifies to just estimating the distribution within the one cluster for which the component density is non-zero.

To perform the clustering step, we must select a particular method of performing clustering with subsets defined by the response variable. Some common classifications of clustering methods include Partitioning-based approaches, Hierarchical approaches, Density-based approaches, Grid-based, Model-based, and Fuzzy-based approaches. In addition, there are some clustering methods suitable for high dimensional data, such as Subspace-based clustering approaches, Hypergraph-based, and concept-based clustering (Mittal et al., 2019).

To ensure the speed of the algorithm and avoid too much data being included in any single cluster, which is easily caused by some methods of clustering, this algorithm uses a hierarchical clustering method. Hierarchical clustering has efficient memory usage, it does not require the storage of a complete distance matrix, which can be computationally expensive for large datasets, and can be tuned to create clusters of specific sizes.

Hierarchical clustering methods are bottom-up clustering methods that begin by forming clusters from individual data points, gradually merging them upward until reaching a unified cluster at the top. Hierarchical clustering methods can be represented using dendrograms, as shown in Figure 5.

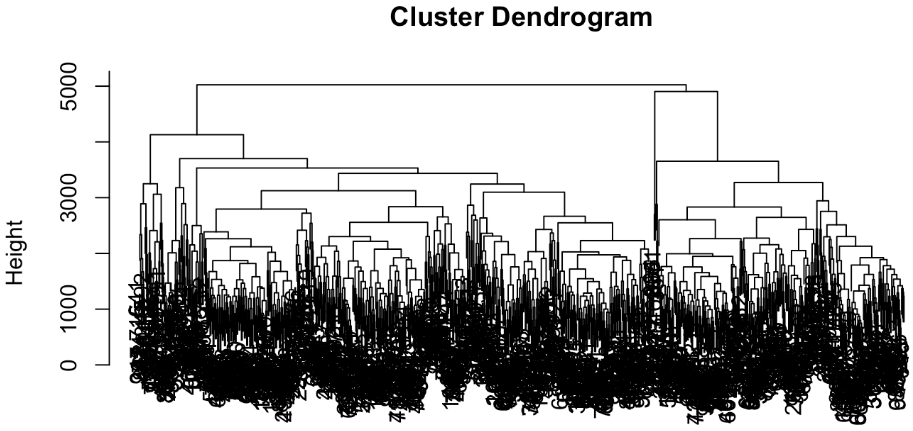


Figure 5

Hierarchical clustering initially treats each data point X as an individual singleton cluster. It then proceeds by computing pairwise distances or similarities between all data points, with Euclidean distances being a commonly utilized metric.

$$D_{ij} = ||X_{i\cdot} - X_{j\cdot}||$$

At each iteration, we identify the two closest clusters based on a specified linkage method and merge them into a single cluster. The choice of linkage method determines how the distance between clusters is calculated. There are 4 types of link functions: Single linkage, Complete linkage, Average linkage, and Centroid linkage. This thesis will use hierarchical clustering with complete linkage which defines the distance between two clusters to be the maximum distance between any two points in the clusters: $f = max(d(x, y))$.

After merging clusters, the distance or similarity matrix is updated to incorporate the distances between the newly formed cluster and the remaining clusters. This process is repeated iteratively until all data points are merged into a single cluster or until a predefined stopping criterion is satisfied.

An advantage of hierarchical clustering is that it doesn't require specifying the number of clusters in advance. Clusters are formed by cutting the dendrogram, and different cutting strategies can yield varying cluster arrangements. Two cutting strategies are commonly employed: setting a cut height and specifying the number of clusters.

1. height of dendrogram: The dendrogram can be pruned by drawing a horizontal line at a chosen height, which reflects the distance between observations and/or clusters. Adjusting height allows dendrograms to form different numbers of clusters.

2. number of clusters: The choice of number of clusters can directly correspond to the number of clusters generated after dendrogram pruning.

After applying clustering to training data, we can form high-similarity clusters. Assuming we know the clusters, we choose the closest clusters as a training dataset which can also maintain high similarity when filtering out some unnecessary data. Given a test point, clustered training dataset, and each cluster's centers, the algorithm is shown below:

Algorithm:

1. Perform clustering within each outcome category level.
2. For a new point, calculate the distance between the test point and the centers of each conditional cluster.
3. Collect training data in the closest cluster for each level of the outcome variable.
4. Performing KNN based on this dataset:
    1. Calculate the distance between test data and collected training data.
    2. Choose the closest k training data.
    3. Predict according to criteria (To maximize accuracy, we usually predict as the category with the highest proportion)

Based on the algorithm, KNN depending on filtered data only needs to use a small proportion of original training data, which reduces a lot of calculation compared to the general KNN method. Even though clustering takes some time to complete, the clustered training data is reusable for many prediction points and can be completed before selecting one or more k values, which is often a very time-intensive activity. This means that applying the KNN classifier on new test data does not require clustering training data again.

Since applying a clustering method before using a KNN classifier tends to remove mostly unnecessary data, the proposed method will approximate KNN with an optimal k value selected. Figure 6 shows the relationship between the k value and model risk of the normal KNN method and the KNN method based on the filtered training data after the hierarchical clustering. Both figures show a similar concave up curve indicating.

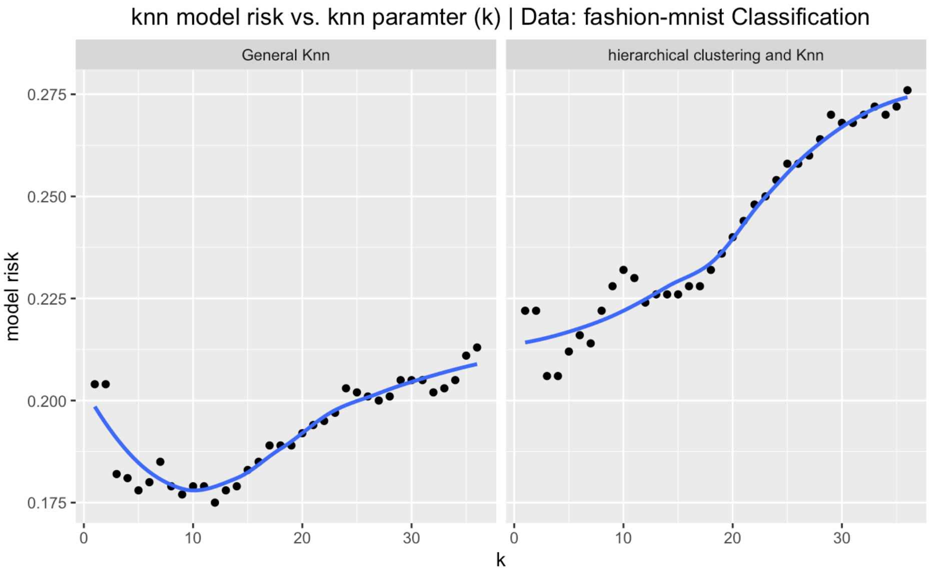


Figure 6

In the next section, the thesis investigates the statistical and computational performance of the proposed method compared to KNN on the full data set.

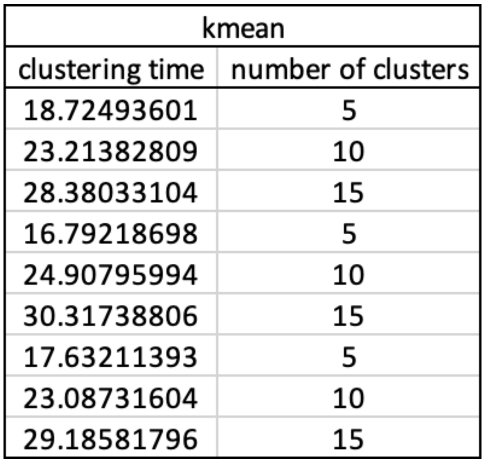
3 Experiments

To evaluate the performance of the proposed method, we conduct experiments using real-world datasets Fashion MNIST, Forest Cover Type Prediction, and Online Shoppers Intention from the UCI Machine Learning repository. The validation process mainly compares the performance of general KNN classifiers and KNN classifiers based on hierarchical clustering filtered training data. What's more, The validation process also includes a comparison of proposed algorithms under different clustering methods.

3.1 Comparing Clustering Methods

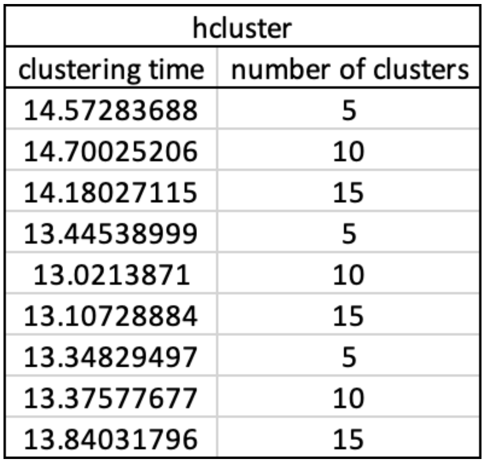
The proposed algorithm employs hierarchical clustering, which draws on flexibility and robustness to noise. This approach enables the utilization of diverse distance metrics and linkage methods, offering adaptability to various data types and clustering goals. Hierarchical clustering techniques tend to exhibit resilience to noise and outliers as they progressively combine or separate clusters based on similarity. This gradual process fosters more reliable clustering outcomes and saves a lot of clustering time. The following table compares applying hierarchical clustering with applying k-man clustering starting with 20 clusters.

In order to ensure the reliability of the comparison, The validation process chose to use 3 different numbers of clusters (5, 10, 15) and repeated the experiment 3 times. The experiment recorded the time used by the k-mean clustering method and the hierarchical clustering on the Fashion MNIST dataset.



kmean	
clustering time	number of clusters
18.72493601	5
23.21382809	10
28.38033104	15
16.79218698	5
24.90795994	10
30.31738806	15
17.63211393	5
23.08731604	10
29.18581796	15

Table 1



hcluster	
clustering time	number of clusters
14.57283688	5
14.70025206	10
14.18027115	15
13.44538999	5
13.0213871	10
13.10728884	15
13.34829497	5
13.37577677	10
13.84031796	15

Table 2

Based on table1 and table 2, the hierarchical clustering method takes less time in the clustering process. In addition, the time required by the K-mean clustering method increases as the number of clusters increases, while the hierarchical clustering method can maintain a relatively stable training time. This is because the hierarchical clustering method uses bottom-up clustering methods and forms clusters through pruning after forming complete dendrograms.

3.2 Computational Performance and Model Accuracy

Utilizing clustering methodology for data filtration on the training dataset proves advantageous in speeding up the training process of the KNN classifier. This approach introduces a new parameter controlling the number of clusters in each class. Therefore, it can control the trade-off between KNN model training time and its accuracy.

Notice that using a single cluster is equivalent to standard KNN. Increasing the number of clusters diminishes the amount of data available for training the KNN classifier. Excessive clustering precipitates a diminution in the cluster's observational pool, thereby impeding the efficacy of the filtered dataset for k-nearest neighbor classification. In Figure 7, I use the Fashion MNIST dataset to compare the impact of choices of different numbers of hierarchical clusters to filter data on the accuracy of the model (model risk is 1- accuracy).

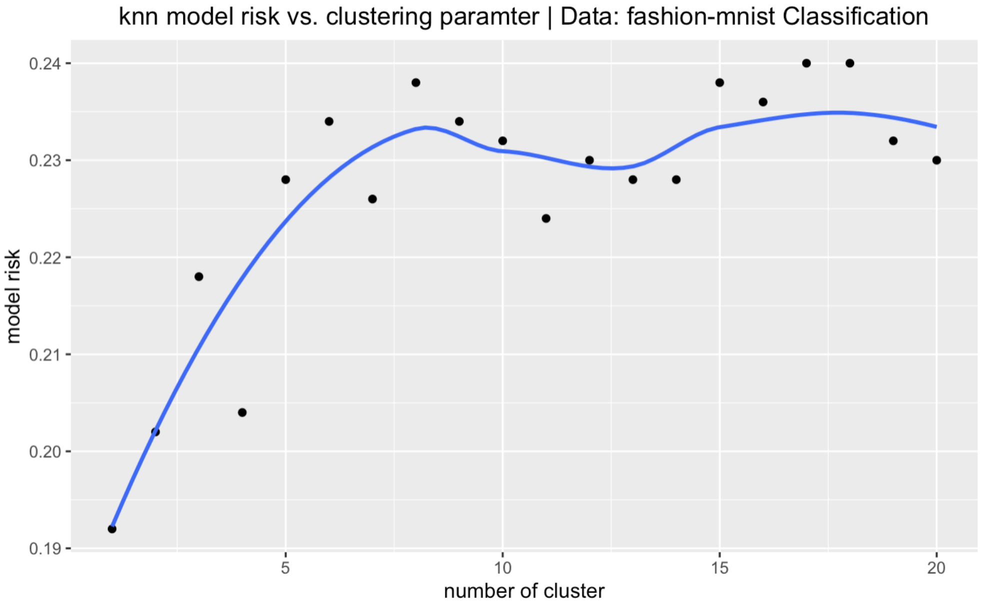


Figure 7

Figure 7 shows a larger number of clusters tends to produce higher model risk. Nonetheless, increasing the number of clusters will allow the KNN classifier to use a smaller training set and reduce training time. Consequently, good selection of the cluster count is important for an optimal balance between computational efficiency and model accuracy. What's more, The parameter for the number of clusters in each class also affects the tuning of the k-nearest neighbor classifier parameters.

3.3 Effect of Dendrogram Cutting

Within the hierarchical clustering methodology, two primary pruning techniques are common: cluster number selection and dendrogram height specification. When opting for cluster number selection, the aim is to ensure an equal distribution of clusters across each class. Conversely, when determining dendrogram height, the objective is to achieve distinct cluster counts for each class. The determination of dendrogram height is contingent upon the inherent characteristics of the data, with the height being Influenced by factors such as Inter-class variance and intra-class similarity.

Based on the two data of the Fashion MNIST dataset and Forest Cover Type Prediction dataset, figure 8-11 compares the relationship between the value of k and model risk generated by using different pruning methods. In order to ensure the reliability of the experiment, 3 different parameters were selected for each pruning method.

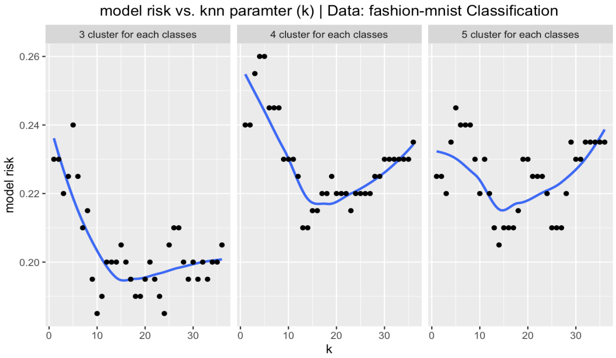


Figure 8

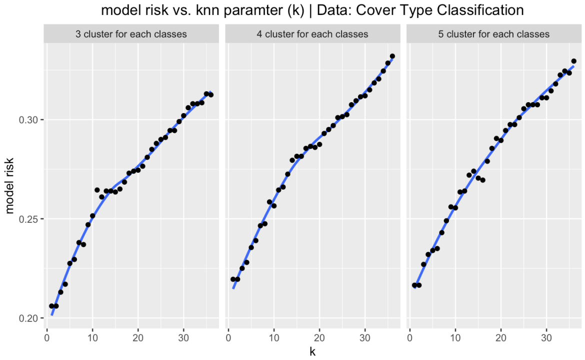


Figure 9

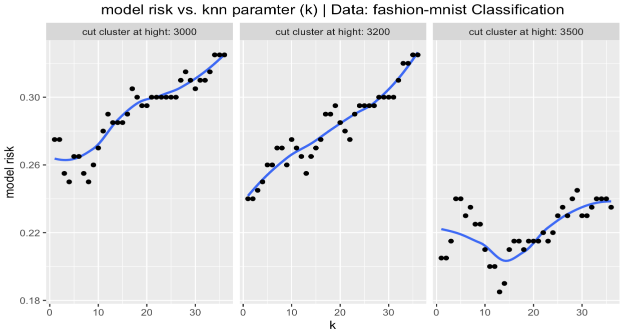


Figure 10

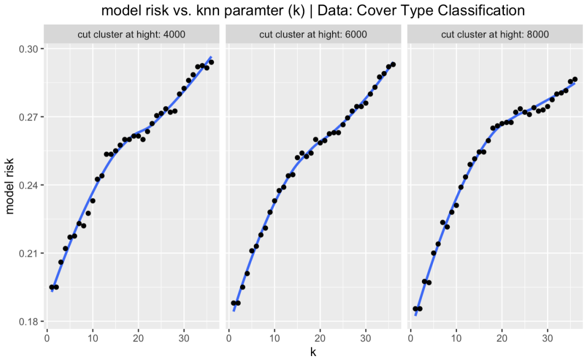
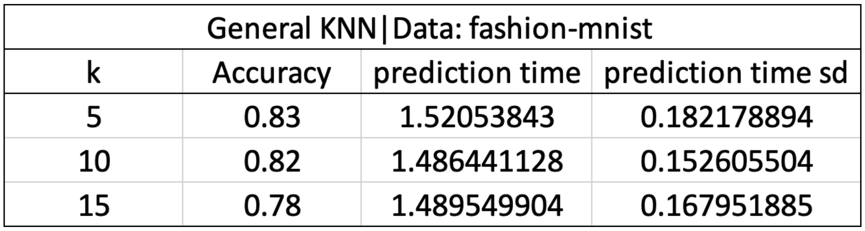


Figure 11

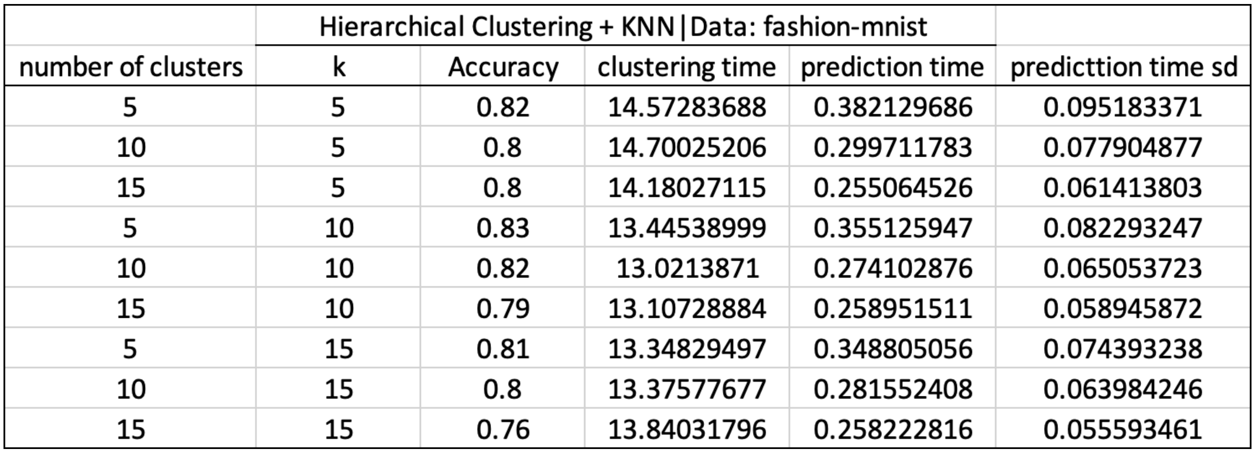
The parameter determining the number of clusters wields considerable influence over the operational effectiveness of the algorithm. It intricately shapes the degree to which the computational efficiency of the k-nearest neighbor classifier is augmented. By and large, the manipulation of this parameter manifests in a discernible pattern: an increase in the number of clusters tends to result In a more stringent filtration of data, consequently facilitating accelerated computational processes. This intricate balance underscores the pivotal role played by the cluster count parameter in optimizing both the fidelity of information retained and the computational expediency of the algorithm. The common cluster numbers are 5, 10, and 15。

Based on the Fashion MNIST dataset, table 3 and table 4 show the comparison between the general KNN method and KNN classifiers using hierarchical clustering filtered training data. The comparison includes the average time it takes to predict each test point (prediction time), the standard deviation of the prediction time (prediction time sd), and model accuracy (Accuracy). In order to ensure the reliability of the comparison, the experiment used 3 different values of k (5,10,15).



General KNN|Data: fashion-mnist			
k	Accuracy	prediction time	prediction time sd
5	0.83	1.52053843	0.182178894
10	0.82	1.486441128	0.152605504
15	0.78	1.489549904	0.167951885

Table 3



	Hierarchical Clustering + KNN|Data: fashion-mnist				
number of clusters	k	Accuracy	clustering time	prediction time	predicttion time sd
5	5	0.82	14.57283688	0.382129686	0.095183371
10	5	0.8	14.70025206	0.299711783	0.077904877
15	5	0.8	14.18027115	0.255064526	0.061413803
5	10	0.83	13.44538999	0.355125947	0.082293247
10	10	0.82	13.0213871	0.274102876	0.065053723
15	10	0.79	13.10728884	0.258951511	0.058945872
5	15	0.81	13.34829497	0.348805056	0.074393238
10	15	0.8	13.37577677	0.281552408	0.063984246
15	15	0.76	13.84031796	0.258222816	0.055593461

Table 4

Experiments show that KNN classifiers using hierarchical clustering filtered training data can still maintain accuracy close to general KNN. What's more, it's worth noting that the proposed algorithm involves two distinct steps. The clustering time, while time-consuming, serves the crucial purpose of partitioning the training data into distinct clusters. Fortunately, this time investment is fixed, meaning no additional time is required when performing prediction. Conversely, the prediction time is remarkably shorter. This step, dedicated to making predictions, benefits significantly from the earlier clustering process. By utilizing the training data to filter out less pertinent information, numerous extraneous calculations are effectively avoided.

Considering the entire set of experiments together, what becomes evident is that employing the data filtered through clustering expedites the model training process significantly compared to using the entire dataset, with minimal compromise to accuracy. This underscores the ability of clustering to exclude training data that holds little similarity with the test point while preserving training data critical for prediction.

3.4 Comparison to Standard KNN

The KNN classifier is renowned for its resemblance to the Bayes classifier. As demonstrated by Stone's theorem, as both k and n approach infinity, and the ratio k/n tends toward 0, the risk of the k-nearest neighbor classifier converges to the Bayes risk (Stone, 1977). These convergence scenarios persist within the proposed algorithm as well when the number of clusters remains fixed.

As the sample size n increases, achieving minimal model risk in the k-nearest neighbor classifier necessitates a larger value for the parameter k. To ensure universal consistency and convergence of model risk, the ratio k/n approaches 0 as both n and k tend toward infinity. Given a fixed number of clusters in each class, the number of observations within each cluster increases proportionally with the sample size n. Consequently, as the sample size increases, the value of k corresponding to the minimum risk in the KNN classifier also increases but at a smaller ratio since k/n approaches 0 while the number of observations within each cluster escalates at the same pace as n. Therefore, there exists a sample size n such that for all sample sizes larger than n, the test point will encompass all necessary points for estimation.

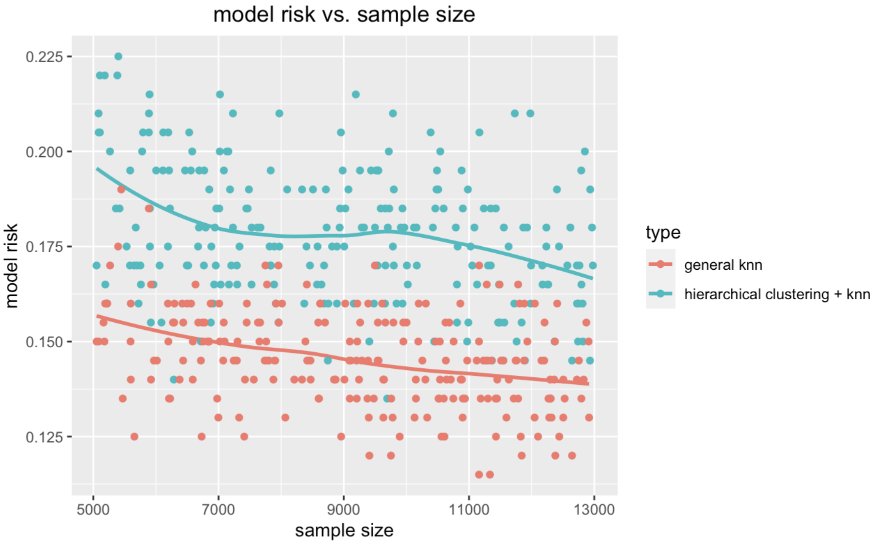


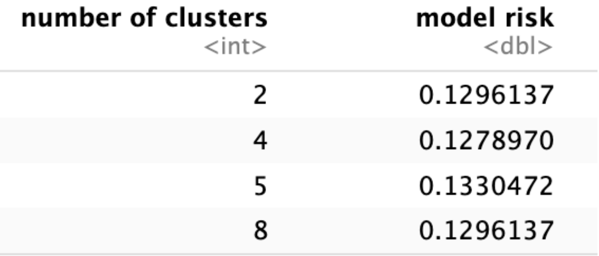
Figure 12

Large numbers may also take a lot of time to run. Figure 12 shows that model risk for the general KNN classifier and KNN classifier using hierarchical clustering filtered training data are both decreased with large sample size at a similar rate. Model risk for the general KNN classifier is always lower, which is because preceding the application of the KNN classifier with data clustering, when the nearest cluster is chosen as the training set, some information within the original training set is inevitably lost which is a common drawback of data filtering approaches. Nonetheless, the essence of clustering lies in its ability to partition the predictor's domain, thereby delineating distinct regions. This partitioning facilitates the implementation of local k-nearest neighbor methods, rendering them both feasible and straightforward. What's even more advantageous is that with the conditional clustered training data, local KNN does not require additional time since filtering data already puts test points in a region.

3.5 Apply Local KNN Classifier

By employing the local KNN classifier on the clustered training dataset, we can fine-tune the KNN algorithm to select a cluster-specific value of k. Table 5 shows the result of applying local KNN to the Online Shoppers Intention dataset.

The experiment clustered the training data conditional on classes through the hierarchical clustering method and applied local KNN in each training data sub-section. In order to ensure the reliability of the experiment, four different numbers of clusters (2, 4, 5, 8) were selected for the experiment. For each number of clusters, the table records the lowest model risk generated by values of k from 1 to 36.

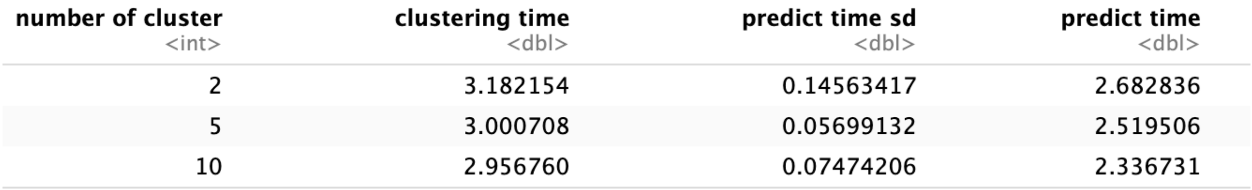


number of clusters	model risk
<int>	<dbl>
2	0.1296137
4	0.1278970
5	0.1330472
8	0.1296137

Table 5

The number of clusters here still plays an important role. It is worth noting that the relationship between the number of clusters and model risk is not strictly increasing. Increasing the number of clusters will shrink the sample size of the training dataset used by KNN leading to larger model risk. However, increasing the number of clusters will partition predictors into more sections, which will give local KNN classifiers more opportunity to have a better balance between variance and bias which leads to lower model risk leading to lower model risk.

Applying the local KNN classifier will not affect the clustering method's ability to reduce running time since applying the local KNN classifier does not require any additional partition of the training dataset. Based on the Online Shoppers Intention dataset with 100 test points and k tuning from 1 to 36, the average time used by the general k-nearest neighbor classifier is 5.055978 seconds with a standard deviation of 0.2974746. For the method of local k-nearest neighbor classifier on clustered training data time used in the clustering step, time used in the prediction step, and its standard deviation are reported below, it is obvious using the clustering method still maintains the ability to reduce running time on prediction even apply local k-nearest neighbor classifier.



number of cluster	clustering time	predict time sd	predict time
<int>	<dbl>	<dbl>	<dbl>
2	3.182154	0.14563417	2.682836
5	3.000708	0.05699132	2.519506
10	2.956760	0.07474206	2.336731

Table 6

4 Conclusion

The KNN classifier is a widely employed method but encounters challenges when confronted with big data due to its sluggish runtime and the extensive range of tuning parameters. This stems from two primary issues. Firstly, KNN necessitates computing pairwise distances between test and training data, which can be computationally taxing, particularly for sizable datasets. Secondly, as the dataset size increases, determining the optimal value of k for enhanced accuracy becomes challenging, as it hinges on the dataset's distribution.

The proposed method in this thesis extends the nearest neighbor classifier algorithm, which categorizes data based on its proximity to neighboring data points. This approach is predicated on the notion that data exhibiting high similarity tends to possess similar conditional distributions of the response variable. By integrating clustering techniques, which group akin data points, we can initially cluster the training data and then utilize the training data within the nearest cluster to classify the test points. Moreover, drawing insights from local KNN, where the k value adjusts based on data point density or x value, facilitates a more adaptable approach. By employing fewer neighbors in low-density regions, we achieve a more balanced bias-variance trade-off. For clustering, a hierarchical clustering method is chosen in this thesis due to its stability and operational efficiency.

Clustering methods naturally yield clusters with fewer observations in low-density regions, aligning seamlessly with the tenets of local KNN. Through empirical testing on real data, it is observed that applying a local k-nearest neighbor classifier on clustered training data aids in mitigating the decline in model accuracy induced by data filtering, all while preserving the efficiency gains attained through clustering.

This thesis also includes a validation process using real data. Experiments have proven that hierarchical clustering can quickly form clusters and the number of clusters will not affect the time consumed by the hierarchical clustering process. In addition, hierarchical clustering can help KNN classifiers train faster without causing a significant loss of model accuracy. The application of Local KNN has also proven to be effective. It can even help the model achieve higher accuracy without losing the ability of clustering to accelerate the training of the model.

Reference

Azadkia, M. (2020).OPTIMAL CHOICE OF k FOR k-NEAREST NEIGHBOR REGRESSION. arXiv.org. https://doi.org/10.48550/arXiv.1909.05495

Cannings, T. I., Berrett, T. B., & Samworth, R. J. (2020). Local nearest neighbour classification with applications to semi-supervised learning. The Annals of Statistics, 48(3). https://doi.org/10.1214/19-aos1868

Cunningham, P., & Delany, S. J. (2021). K-nearest neighbour classifiers - a tutorial. ACM Computing Surveys, 54(6), 1–25. https://doi.org/10.1145/3459665

Deng, Z., Zhu, X., Cheng, D., Zong, M., & Zhang, S. (2016). Efficient KNN classification algorithm for Big Data. Neurocomputing, 195, 143–148. https://doi.org/10.1016/j.neucom.2015.08.112

Hastie, T., Tibshirani, R., & Friedman, J. H. (2023). The elements of Statistical Learning: Data Mining, Inference, and prediction (second edition). Shi jie tu shu chu ban gong si.

Hart, P. (1968). The condensed nearest neighbor rule (corresp.). IEEE Transactions on Information Theory, 14(3), 515–516. https://doi.org/10.1109/tit.1968.1054155

International Business Machines Corporation. (n.d.-a). Usage of KNN. usage of k-Nearest Neighbors (KNN). https://www.ibm.com/docs/en/ias?topic=knn-usage

Korn, F., Sidiropoulos, N., Faloutsos, C., Siegel, E., & Protopapas, Z. (1996). Fast and effective similarity search in medical tumor databases using morphology. In SPIE Proceedings (Vol. 2916, pp. 116-129). https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=c6b0508446947f9ed3f26fcdac67d77e1309f3dc

Mittal, M., Goyal, L. M., Hemanth, D. J., & Sethi, J. K. (2019). Clustering approaches for high‑dimensional databases: A Review. WIREs Data Mining and Knowledge Discovery, 9(3). https://doi.org/10.1002/widm.1300

Saadatfar, H., Khosravi, S., Hassannataj Joloudari, J., Mosavi, A., & Shamshirband, S. (2020). A new K-nearest neighbors classifier for big data based on efficient data pruning. Mathematics, 8(2), 286. https://doi.org/10.3390/math8020286

Stone, C. J. (1977). Consistent nonparametric regression. The Annals of Statistics, 5(4). https://doi.org/10.1214/aos/1176343886

Susheela Devi, V., & Narasimha Murty, M. (2002). An incremental prototype set building technique. Pattern Recognition, 35(2), 505–513. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=6029643ee549753f8702071fc8bf7fba98c7acf3